Infinite Networks, Halting and Local Algorithms (Draft) *

Antti Kuusisto
University of Tampere
Finland

July 31, 2013

Abstract

The immediate past has witnessed an increasing amount of interest in local algorithms, i.e., constant time distributed graph algorithms. In the recent survey of local algorithms (Suomela, ACM Computing Surveys, 2013), it is argued that local algorithms provide a framework that could be used in order to (theoretically) control infinitely large networks in finite time. We establish for a comprehensive collection of models of distributed computing that if infinite networks are included in the class of structures investigated, then every halting distributed algorithm is in fact a local algorithm. The models studied include various weak models of distributed computing in anonymous networks investigated by Hella and co-authors in (Hella et al., PODC 2012).

1 Introduction

This work is a study of deterministic distributed algorithms for arbitrary networks, including infinite structures in addition to finite ones. In the recent survey article [14], Suomela points out that distributed constant-time algorithms are a reasonable choice for (theoretically) controlling infinitely large networks in finite time. In this article we show that for a rather comprehensive collection of models of distributed computing, constant-time algorithms are in a sense the only choice. We define a framework—based on a class of message passing automata and relational structures—that contains a comprehensive variety of models of distributed computation in anonymous networks, i.e., networks without ID-numbers. We then show that all halting algorithms definable in this framework are in fact local algorithms, i.e., distributed constant-time algorithms.

There are several fields of study where the objects of investigation could rather naturally be regarded as infinite distributed (anonymous) communication networks. Cellular automata (see [1]) provide perhaps the most obvious and significant example of such structures. But of course there are others. Crystal lattices and the brain, for example, are massive network systems often modelled by infinite structures. It therefore makes sense to investigate distributed computation models in frameworks with infinite structures in addition to finite ones.

The widely studied *port-numbering model* (see [2, 11, 12]) occupies a central stage in the investigations below. This model can be directly extended to the framework containing infinite structures. In the port-numbering model, a node of degree $k \leq n$, where n is a globally known finite degree bound, receives messages through k input ports and sends messages through k output ports. The processors in the nodes can send different messages to different

*ISBN: 978-951-44-9192-4

neighbours, and also see from which port incoming messages arrive. There are no ID-numbers in this framework, which is in a sense a well-justified choice when infinite models are included in the picture: in the most obvious choices for the theoretical modelling of computation in infinite networks, even the reading of local ID:s would take infinitely long, so (at least synchronized) communication using ID-numbers would be impossible. Notice also that for example cellular automata can be directly simulated in the port-numbering model.

Below we define a rather general and flexible distributed computing model based on relational structures and synchronized message passing automata. For example, each one of the models of distributed computing studied in [11], including the port-numbering model $\mathrm{VV_c}$, can be directly simulated in our framework by restricting attention to suitable classes of structures and automata. One of our principal results, Theorem 4.3, states that if the class of communication networks studied is definable by a sentence of *first-order logic*, then all halting algorithms are local algorithms. For example the classes of networks for the VV_c model are easily seen to be definable by first-order formulae, *as long as structures are allowed to be infinite*. In fact, all classes of structures for the models studied in [11] can easily be seen to be first-order definable when the requirement of finiteness is lifted.

The proof of Theorem 4.3 employs methods offered by mathematical logic, and extends the work begun in [11] (see also the extended version [12]). The article [11] extends the scope of *descriptive complexity theory* (see [10, 9, 13]) to the realm of distributed computing by identifying a highly canonical one-to-one link between *local* algorithms and formulae of modal logic (see [4, 5, 7]). We first establish that *every* halting algorithm in our computation model can be specified by a pair of (possibly infinite) formulae of modal logic. We then use this realization, together with a well-known link between modal logic and first-order logic, in order to facilitate the use of the *compactness theorem* (see [9]), which is the last step in our proof.

We also investigate the mathematical phenomena that lead to halting in the finite. The weakest (in computational capacity) of the models investigated in [11] is the SB model, which corresponds to the port-numbering model, but with the additional twist that processors are in fact not allowed to see individual port numbers at all. We investigate a generalized version of this model, and show that even in the very weak resulting framework, there are algorithms that are halting but non-local in the finite. This is by no means trivial, and is in fact one of our two principal results, the other one being Theorem 4.3. In order to prove the result, we employ tools from combinatorics or words, namely, the infinite *Morse-Thue sequence*. This infinite binary sequence is known to be cube-free, i.e., it does not have a prefix of the type $tuuu$, where u is a nonempty word. This lack of periodicity allows us to design an algorithm that is halting but non-local in the investigated computationally weak framework, as long as attention is limited to finite structures.

The current article continues the logic based investigations into the theory of distributed computing initiated in [11]. There are several possible related future research directions that could and should be followed. The article [11] and the current article investigate anonymous models by using tools of modal logic. *Hybrid logic* (see [3]) is a subfield of modal logic that investigates systems extended with so-called *nominals*, which are objects that are true in exactly one node of each model, and therefore naturally correspond to ID-numbers. It would be interesting to see how this link could be used.

This article is a draft. A polished up and corrected version will appear somewhere later on.

2 Preliminaries

Let Π be a finite set of *unary predicate symbols* $P \in \Pi$ and $\mathcal{R}$ a finite set of *binary predicate symbols* $R \in \mathcal{R}$. These symbols are also called *relation symbols*. The set of $(\Pi, \mathcal{R})$-*formulae* of *modal logic* $\mathrm{ML}(\Pi, \mathcal{R})$ is generated by the grammar

$$\varphi \ ::= \ \top \mid P \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \langle R \rangle \varphi,$$

where P is any symbol in Π, R any symbol in $\mathcal{R}$, and $\top$ is a logical constant symbol. Let $\mathrm{VAR} = \{ x_i \mid i \in \mathbb{N} \}$ be a set of *variable symbols*. The set of $(\Pi, \mathcal{R})$-*formulae* of *first-order logic* $\mathrm{FO}(\Pi, \mathcal{R})$ is generated by the grammar

$$\varphi \ ::= \ \top \mid x = y \mid P(x) \mid R(x, y) \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \exists x \, \varphi,$$

where x and y are any symbols in VAR, P any symbol in Π, R any symbol in $\mathcal{R}$, and $\top$ a logical constant symbol. For both logics, we define the abbreviation $\bot = \neg\top$. We also use the abbreviation symbols $\vee$, $\rightarrow$ and $\leftrightarrow$ in the usual way. The *modal depth* $md(\varphi)$ of a formula is defined recursively such that $md(\top) = md(P) = 0$, $md(\neg\,\psi) = md(\psi)$, $md(\psi \wedge \chi) = max\{md(\psi), md(\chi)\}$, and $md(\langle R \rangle\, \psi) = md(\psi) + 1$.

Let $\Pi = \{P_1, ..., P_n\}$ and $\mathcal{R} = \{R_1, ..., R_m\}$. A $(\Pi, \mathcal{R})$-*model* is a structure $M = (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M)$, where W is an arbitrary nonempty set (called the *domain* of the model M), each P_i^M is a unary relation $P_i^M \subseteq W$, and each R_i^M a binary relation $R_i^M \subseteq W \times W$. The semantics of $\mathrm{ML}(\Pi, \mathcal{R})$ is defined with respect to *pointed* $(\Pi, \mathcal{R})$-*models* (M, w), where $M = (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M)$ is a $(\Pi, \mathcal{R})$-model and $w \in W$ a *point* or a *node* of (the domain of) M.

For each $P_i \in \Pi$, we define $(M, w) \models P_i$ iff $w \in P_i^M$. We also define $(M, w) \models \top$. We then recursively define

$$\begin{array}{lcl}
(M, w) \models \neg\,\varphi & \Leftrightarrow & (M, w) \not\models \varphi, \\
(M, w) \models (\varphi \wedge \psi) & \Leftrightarrow & (M, w) \models \varphi \text{ and } (M, w) \models \psi, \\
(M, w) \models \langle R_i \rangle\, \varphi & \Leftrightarrow & \exists v \in W \big((w, v) \in R_i^M \text{ and } (M, v) \models \varphi \big).
\end{array}$$

The semantics of $\mathrm{FO}(\Pi, \mathcal{R})$ is defined with respect to $(\Pi, \mathcal{R})$-*interpretations* (M, f), where $M = (W, P_1^M, ..., P_n^M, R_1^M, ..., R_m^M)$ is a $(\Pi, \mathcal{R})$-model and f is an *assignment function* $f : \mathrm{VAR} \longrightarrow W$ giving an interpretation to each one of the variables in VAR. We define that $(M, f) \models x = y \Leftrightarrow f(x) = f(y)$, $(M, f) \models P_i(x) \Leftrightarrow f(x) \in P_i^M$, and $(M, f) \models R_i(x, y) \Leftrightarrow (f(x), f(y)) \in R_i^M$. We also define $(M, f) \models \top$. We then recursively define

$$\begin{array}{lcl}
(M, f) \models \neg\,\varphi & \Leftrightarrow & (M, f) \not\models \varphi, \\
(M, f) \models (\varphi \wedge \psi) & \Leftrightarrow & (M, f) \models \varphi \text{ and } (M, f) \models \psi, \\
(M, f) \models \exists x \varphi & \Leftrightarrow & \exists v \in W \big((M, f[x \mapsto v]) \models \varphi \big),
\end{array}$$

where $f[x \mapsto v]$ is the function $g : \mathrm{VAR} \to W$ such that

$$g(y) = \begin{cases} v & \text{if } y = x, \\ f(y) & \text{if } y \neq x. \end{cases}$$

It is important to notice that modal logic can be directly translated into first-order logic. This is folklore to modal logicians, but worth discussing explicitly in the context of the current article. We define the *standard translation* from $\mathrm{ML}(\Pi, \mathcal{R})$ into $\mathrm{FO}(\Pi, \mathcal{R})$. Let $St_x(\top,) = \top$, $St_x(P_i) = P_i(x)$, $St_x(\,(\varphi \wedge \psi)\,) = \big(St_x(\varphi) \wedge St_x(\psi)\big)$, $St_x(\neg\varphi) = \neg St_x(\varphi)$, and $St_x(\langle R_i \rangle \varphi) = \exists y \big(R_i(x, y) \wedge St_y(\varphi)\big)$, where $y \neq x$. It is easy to see that $(M, v) \models \varphi$

iff $(M, f[x \mapsto v]) \models St_x(\varphi)$. Therefore modal logic can simply be considered a *fragment* of first-order logic.

If Φ is a set of formulae, $\bigvee \Phi$ and $\bigwedge \Phi$ denote the *disjunction* and *conjunction* of the formulae in Φ. The set Φ can be infinite, but then neither the formula $\bigvee \Phi$ nor the formula $\bigwedge \Phi$ is a formula of $\mathrm{ML}(\Pi, \mathcal{R})$ or $\mathrm{FO}(\Pi, \mathcal{R})$. Let X be a pointed model or an interpretation. We define $X \models \bigvee \Phi$ if there exists at least one formula $\varphi \in \Phi$ such that $X \models \varphi$. We define $X \models \bigwedge \Phi$ if $X \models \varphi$ for all $\varphi \in \Phi$. If T is a set of formulae, then $X \models T$ means that $X \models \varphi$ for all $\varphi \in T$. When we write $T \models \varphi$, we mean that the implication $X \models T \Rightarrow X \models \varphi$ holds for all pointed $(\Pi, \mathcal{R})$-models (or $(\Pi, \mathcal{R})$-interpretations) X. Two $(\Pi, \mathcal{R})$-formulae φ and ψ are *equivalent*, if the equivalence $X \models \varphi \Leftrightarrow X \models \psi$ holds for all pointed $(\Pi, \mathcal{R})$-models (or all $(\Pi, \mathcal{R})$-interpretations) X.

Let $\mathcal{H}$ be a class of pointed $(\Pi, \mathcal{R})$-models, and let $\mathcal{K} \subseteq \mathcal{H}$. A modal formula φ *defines* the class $\mathcal{K}$ with respect to $\mathcal{H}$, if for all $(M, w) \in \mathcal{H}$, we have $(M, w) \models \varphi \;\Leftrightarrow\; (M, w) \in \mathcal{K}$. If some formula ψ defines a class $\mathcal{J}$ of pointed $(\Pi, \mathcal{R})$-models with respect to the class of exactly all pointed $(\Pi, \mathcal{R})$-models, we simply say that ψ defines $\mathcal{J}$. The set $Fr(\varphi)$ of *free variables* of a first-order $(\Pi, \mathcal{R})$-formula φ is defined recursively such that $Fr(\top) = \emptyset$, $Fr(P(x)) = \{x\}$, $Fr(x = y) = Fr(R(x, y)) = \{x, y\}$, $Fr(\varphi \wedge \psi) = Fr(\varphi) \cup Fr(\psi)$, $Fr(\neg\varphi) = Fr(\varphi)$, and $Fr(\exists x\varphi) = Fr(\varphi) \setminus \{x\}$. A first-order $(\Pi, \mathcal{R})$-formula whose set of free variables is empty, is a first-order $(\Pi, \mathcal{R})$-*sentence*. If M is a $(\Pi, \mathcal{R})$-model and φ a first-order $(\Pi, \mathcal{R})$-*sentence*, we write $M \models \varphi$ if $(M, f) \models \varphi$ for some assignment f. Let $\mathcal{J}$ be any class of pointed $(\Pi, \mathcal{R})$-models. A first-order $(\Pi, \mathcal{R})$-*sentence* φ *defines* the class $\mathcal{J}$ of pointed $(\Pi, \mathcal{R})$-models if for all pointed $(\Pi, \mathcal{R})$-models (M, w), we have $M \models \varphi \Leftrightarrow (M, w) \in \mathcal{J}$.

Let Π and $\mathcal{R} = \{ R_1, ..., R_k \}$ be finite sets of unary predicate symbols and binary predicate symbols, respectively. A *message passing automaton* A of the vocabulary $(\Pi, \mathcal{R})$, or a $(\Pi, \mathcal{R})$-automaton, is a tuple $(Q, \mathcal{M}, \pi, \delta, \mu, F, G)$. The object Q is a nonempty set of *states*. The set Q can be finite or countably infinite. The object $\mathcal{M}$ is a nonempty set of *messages*. The set $\mathcal{M}$ can be finite or countably infinite. The object $\pi : Pow(\Pi) \longrightarrow Q$ is an *initial transition function* that determines the beginning state of A. The object $\delta : (Pow(\mathcal{M}))^k \times Q \longrightarrow Q$ is a *transition function* that constructs a new state in Q when given a k-tuple $(N_1, ..., N_k) \in (Pow(\mathcal{M}))^k$ of received message sets and a previous state in Q. The object $\mu : Q \times \mathcal{R} \longrightarrow \mathcal{M}$ is a *message construction function* that constructs a message for the automaton to send forward when given a state of the automaton and a communication channel R_i. The object $F \subseteq Q$ is the set of *accepting states* of the automaton. The object $G \subseteq Q \setminus F$ is the set of *rejecting states* of the automaton.

Let $\mathcal{R} = \{ R_1, ..., R_k \}$ and $\Pi = \{ P_1, ..., P_m \}$. If (M, w) is a $(\Pi, \mathcal{R})$-model, the set of R_i-*predecessors* of w is the set of nodes u in the domain of M such that $R_i(u, w)$, and the set of R_i-*successors* of w is the set of nodes u such that $R_i(w, u)$. The set of R_i-successors of w is denoted by $succ(R_i, w)$. A message passing $(\Pi, \mathcal{R})$-automaton A is *run* on a $(\Pi, \mathcal{R})$-model $M = \big(W, R_1, ..., R_k, P_1, ..., P_m\big)$, considered to be a distributed system. On the intuitive level, we place a copy (A, w) of the automaton to each node $w \in W$. Then, each automaton (A, w) first scans the *local information* of the node w, i.e., finds the set of predicates $P_i \in \Pi$ such that $(M, w) \models P_i$, and then makes a transition to a beginning state based on the local information. Then, the automata (A, w), where $w \in W$, begin running in *synchronized steps*. During each step, each automaton (A, w) sends, for each $i \in \{ 1, ..., k \}$, a message m_i to the R_i-*predecessors* of w.[1] The automaton (A, w) also receives a tuple $(N_1, ..., N_k)$ of message sets N_i such that set N_i is received from the R_i-successors of w. Then the automaton updates

[1] Therefore information flows opposite to the direction of the arrows (i.e., ordered pairs) of R_i. The reason for this choice is technical, and could be avoided. The choice is due to the relationship between modal logic and message passing automata. An alternative approach would be to consider modal logics with the truth of $\langle R_i \rangle \varphi$ defined such that $(M, w) \models \langle R_i \rangle \varphi$ iff $\exists v \in W\big((v, w) \in R^M$ and $(M, v) \models \varphi \big)$.

its state based on the received messages and the previous state.

More formally, a $(\Pi, \mathcal{R})$-model $\big(W, R_1, ..., R_k, P_1, \ ..., P_m\big)$ and a $(\Pi, \mathcal{R})$-automaton $A = \big(Q, \mathcal{M}, \pi, \delta, \mu, F, G\big)$ define a synchronized distributed computation system which executes *communication rounds* defined as follows. Each round $n \in \mathbb{N}$ defines a *global configuration* $f_n : W \longrightarrow Q$. The configuration f_0 of the zeroth round is the function f_0 such that $f_0(w) = \pi(\{\ P \in \Pi \mid w \in P^M\ \})$ for all $w \in W$. Recursively, assume that we have defined f_n, and let $(N_1, ..., N_k)$ be a tuple of message sets

$$N_i \ = \ \big\{\ m \in \mathcal{M} \mid m = \mu(f_n(v), R_i),\ v \in \mathit{succ}(R_i, w)\ \big\}.$$

Then $f_{n+1}(w) \ = \ \delta\big(\,(N_1, ..., N_k),\ f_n(w)\,\big)$.

When we talk about *the state of the automaton A at the node w in round n*, we mean the state $f_n(w)$. We define that an automaton A *accepts* a pointed model (M, w) if there exists some $n \in \mathbb{N}$ such that $f_n(w) \in F$, and furthermore, for all $m < n$, $f_m(w) \notin G$. Similarly, A *rejects* (M, w) if there exists some $n \in \mathbb{N}$ such that $f_n(w) \in G$, and for all $m < n$, $f_m(w) \notin F$. Notice that A may keep passing messages and changing state even after it has accepted or rejected. Defining automata this way was an essential step in the original process of developing the results below and those in [6]. Also, this definition is rather general, and automata that stop sending messages can be modelled in this framework by for example automata that begin sending some constant message "I have halted" once they have accepted or rejected.

Let $\mathcal{C}$ be the class of all pointed $(\Pi, \mathcal{R})$-models. Let $\mathcal{H} \subseteq \mathcal{C}$. We say that A accepts (rejects) $\mathcal{H}$, if the class of pointed models in $\mathcal{C}$ that A accepts (rejects) is $\mathcal{H}$. Let $\mathcal{J} \subseteq \mathcal{K} \subseteq \mathcal{C}$. We say that A accepts (rejects) $\mathcal{J}$ in $\mathcal{K}$, if the class of pointed models in $\mathcal{K}$ that A accepts (rejects) is $\mathcal{J}$. A $(\Pi, \mathcal{R})$-automaton A *converges* in the class $\mathcal{K}$, if for all $(M, w) \in \mathcal{K}$, the automaton A either accepts or rejects (M, w). A $(\Pi, \mathcal{R})$-automaton $A = \big(Q, \mathcal{M}, \pi, \delta, \mu, F, G\big)$ *halts* in $\mathcal{K}$ if A converges in $\mathcal{K}$, and furthermore, for each state $q \in F \cup G$ that is obtained by A at some $(M, w) \in \mathcal{K}$, the state of A at (M, w) will be q forever once q has been once obtained. We say that the automaton A *specifies a local algorithm* in $\mathcal{K}$ if there exists some $n \in \mathbb{N}$ such that for all $(M, w) \in \mathcal{K}$, the automaton A accepts or rejects (M, w) in some round $m \leq n$. For the sake of curiosity, note that even if A specifies a local algorithm, it does not necessarily halt. But of course a corresponding halting automaton exists.

Our framework with $(\Pi, \mathcal{R})$-automata operating on $(\Pi, \mathcal{R})$-models is quite flexible and general. For example, each one of the models of distributed computing studied in [11] can be directly simulated in our framework by restricting attention to suitable classes of structures and automata. Let us have a closer look at this matter.

Let $\mathcal{R} = \{R\}$ and let Π be any finite set. If M is $(\Pi, \mathcal{R})$-model, where R^M is a symmetric and irreflexive binary relation, then let us call M an SB(Π)*-model*. These models are intimately related to the weakest (in computational capacity) computation model SB studied in [11].

Let $n \in \mathbb{N} \setminus \{0\}$ and $S = \{\,1, ..., n\,\}$. Let $\Pi = \{\,P_0, ..., P_n\,\}$, and $\mathcal{R} = \{\,R_{(i,j)} \mid (i, j) \in S \times S\,\}$. A pointed $(\Pi, \mathcal{R})$-model (M, w) is an *n-port-numbering structure*, or a PN(n)-structure, if it satisfies the following rather long list of conditions. The union R of the relations $R^M_{(i,j)}$ is a symmetric and irreflexive relation. For each two distinct pairs $(i, j), (k, l) \in S \times S$, if $R^M_{(i,j)}(u, v)$, then $R^M_{(k,l)}(u, v)$ does not hold. For each $(i, j) \in S \times S$, if $R^M_{(i,j)}(u, v)$, then $R^M_{(j,i)}(v, u)$. For each $(i, j) \in S \times S$, the out-degree and in-degree of $R^M_{(i,j)}$ is at most one at each node. If $R_{(i,j)}(u, v)$ for some nodes u and v and some $i, j \in S$, then, if $k < i$, there exists some $l \in S$ and some node v' such that $R^M_{(k,l)}(u, v')$. Similarly, if $R^M_{(i,j)}(u, v)$ for some nodes u and v and some $i, j \in S$, then, if $k < j$, there exists some $l \in S$ and some node u' such that $R^M_{(l,k)}(u', v)$. Finally, for each node u and each $i \in S$, we have $u \in P^M_i$ if and only

if the out-degree (or equivalently, in-degree) of the union R of all the relations $R^M_{(l,j)}$ is i at u.

It is straightforward to show that there exists a first-order $(\Pi, \mathcal{R})$-sentence $\varphi_{\mathrm{PN}(n)}$ that defines the class $\mathcal{PN}(n)$ of all PN(n)-structures. The class of *finite* PN(n)-structures is exactly the collection of communication networks of maximum degree n used in computations in the framework of the port-numbering model VV_c of [11, 12]. The related collection of VV_c algorithms *corresponds to* the class of algorithms that can be specified by $(\Pi, \mathcal{R})$-automata that halt in all finite PN(n) structures. Therefore the class $\mathcal{PN}(n)$ of exactly all PN(n)-structures, together with $(\Pi, \mathcal{R})$-automata, defines a generalization of the port-numbering model to the context with infinite structures in addition to finite ones. Theorem 4.3 shows that all halting algorithms for $\mathcal{PN}(n)$ run in constant-time. There are no non-local halting algorithms in the framework of the port-numbering model when infinite structures are included in the picture.

3 Halting in the finite

When attention is restricted to finite models only, halting non-local algorithms exist even in the framework of SB(Π)-models. Let $\Pi = \{ P_0, P_1, Q_1, Q_2, Q_3 \}$ and $\mathcal{R} = \{R\}$. We shall show that there exists a non-local algorithm that halts in the class of exactly all finite SB(Π)-models.

We shall begin by sketching an intuitive picture of the way the algorithm roughly works. The unary predicates P_0 and P_1 will be used in order to define binary words in $\{ 0, 1\}^*$ that correspond to finite paths[2] in $(\Pi, \mathcal{R})$-models. Each node w will store sets of increasingly long finite binary words that are generated along paths that originate from w. The related paths will be oriented by the predicates Q_1, Q_2 and Q_3 such that if a node u is labelled by Q_i, then its successor is labelled by $Q_{f(i)}$, where $f : \{ 1, 2, 3 \} \longrightarrow \{ 1, 2, 3 \}$ is the cyclic permutation $1 \mapsto 2 \mapsto 3 \mapsto 1$. A node w will halt if it records some word $s \in \{ 0, 1 \}^*$ that contains a *cube* as a factor, i.e., a word $s = tuuuv$, where u is a *nonempty* word in $\{ 0, 1 \}^*$ and $t, v \in \{ 0, 1 \}^*$. Upon halting, the node will send an instruction to halt to its neighbours, who will then pass it on and halt, so the halting instruction spreads out in the connected component of w causing further and further nodes to halt. A globally spreading halting instruction can also be generated due to the detection of an undesirable labelling pattern defined by the unary predicates in Π. A node accepts iff it halts in a round $n \in \mathbb{N}$ for some even number n. Otherwise it will reject. The fact that the algorithm is not local follows from the existence of arbitrarily long cube-free finite words. Indeed, there exists an infinite cube-free word, known as the *Morse-Thue sequence*, see [1] for example.

Let us say that a node w is a Q_1-*node* if $(M, w) \models Q_1 \wedge \neg Q_2 \wedge \neg Q_3$. Similarly, w is a Q_2-node if $(M, w) \models Q_2 \wedge \neg Q_1 \wedge \neg Q_3$ and a Q_3-*node* if $(M, w) \models Q_3 \wedge \neg Q_1 \wedge \neg Q_2$. A node w is *properly oriented* if w is a Q_i-node for some $i \in \{1, 2, 3\}$, and furthermore, w has a Q_j-node as a neighbour if and only if $j \in \{1, 2, 3\} \setminus \{i\}$. A node w is *proper* if it is properly oriented, and furthermore, either $(M, w) \models P_0 \wedge \neg P_1$ or $(M, w) \models P_1 \wedge \neg P_0$.

Let $\{ 0, 1 \}^+$ denote the set $\{ 0, 1 \}^* \setminus \{ \lambda \}$, where λ is the empty word. Let $\mathcal{L}$ be the set of finite subsets of $\{ 0, 1 \}^+$. The set of states of the automaton A defining our algorithm is $\mathcal{L} \times \{ 0, 1 \} \times \{ 1, 2, 3 \} \times \{ \mathit{run}, \mathit{halt} \} \times \{ 0, 1\}$, plus an additional finite set H of states. The set of messages is $\mathcal{L} \times \{ 1, 2, 3 \} \times \{ \mathit{run}, \mathit{halt} \}$, plus an additional finite set H' of messages.

The first object S_1 of a state $(S_1, S_2, S_3, S_4, S_5)$ of a node w in round n encodes a collection of words corresponding to paths originating from w. The paths are labelled by Q_1, Q_2 and

[2] A path here is any function from some initial segment of $\mathbb{N}$ to the domain of the graph such that $f(i)$ and $f(i + 1)$ are connected by an edge.

Q_3 in a cyclic fashion. The longer the automaton computes, the longer the words in S_1 get. S_2 encodes the symbol $P \in \{P_0, P_1\}$ such that $(M, w) \models P$. S_3 encodes the symbol $Q \in \{Q_1, Q_2, Q_3\}$ such that $(M, w) \models Q$. A state $(S_1, S_2, S_3, S_4, S_5)$ is an accepting state if $S_4 = \mathit{halt}$ and $S_5 = 0$. A state $(S_1, S_2, S_3, S_4, S_5)$ is a rejecting state if $S_4 = \mathit{halt}$ and $S_5 = 1$.

The set S_1 of a message (S_1, S_2, S_3) is a set of binary words, corresponding to the language recorded by the sending node. S_2 encodes the label in $\{Q_1, Q_2, Q_3\}$ that labels the sending node. S_3 is a halting instruction if $S_3 = \mathit{halt}$.

In the very beginning of the computation, the algorithm makes use of the the additional states in H and messages in H' in order to establish whether the nodes in the network are proper. Then, if a node w is proper and $(M, w) \models P_x \wedge Q_y$, where $x \in \{0, 1\}$ and $y \in \{1, 2, 3\}$, the state of A at w in round 1 is $(\{x\}, x, y, \mathit{run}, 1)$. If w is not proper, then the state of A at w in round 1 is $(\{x\}, x, y, \mathit{halt}, 1)$, where x and y are fixed arbitrarily.

Let S be the set of messages received by a node w in some round $n+1$, where $n \in \mathbb{N} \setminus \{0\}$. Let $(S_1, S_2, S_3, S_4, S_5)$ be the state of w in round n. If $S_4 = \mathit{halt}$, then the new state is $(S_1, S_2, S_3, S_4, S_5)$. Otherwise the new state $(S'_1, S'_2, S'_3, S'_4, S'_5)$ is defined as follows.

Let $f : \{1, 2, 3\} \longrightarrow \{1, 2, 3\}$ be the cyclic permutation $1 \mapsto 2 \mapsto 3 \mapsto 1$. Assume first that S does not contain a tuple of the form (A, B, halt). Then we define

$$S'_1 = \{\ v \in \{0, 1\}^* \mid v = xu \text{ such that } x = S_2 \text{ and } u \in T \text{ for some } (T, f(S_3), \mathit{run}) \in S\ \}.$$

We set $S'_2 = S_2$ and $S'_3 = S_3$. We let $S'_4 = \mathit{halt}$ iff S'_1 contains a word with a cube as a factor. We let $S'_5 \in \{0, 1\} \setminus \{S_5\}$. If S contains a tuple of the form (A, B, halt), we define $(S'_1, S'_2, S'_3, S'_4, S'_5) = (X, Y, Z, \mathit{halt}, x)$, where $x \in \{0, 1\} \setminus \{S_5\}$, and X, Y and Z are fixed arbitrarily.

Let $(S_1, S_2, S_3, S_4, S_5)$ be the state of A at w in round n, where $n \in \mathbb{N} \setminus \{0\}$. If $S_4 = \mathit{run}$, the message broadcast by A at w in round $n+1$ is (S_1, S_3, run), and if $S_4 = \mathit{halt}$, the message is (X, Y, halt), where X and Y are fixed arbitrarily.

Theorem 3.1. *Let* Π *be as defined above. There exists an* SB(Π) *automaton A that is halting but non-local in the class of finite pointed* SB(Π)*-models. There exists no* SB(Π) *automaton that specifies a local algorithm in the finite and accepts exactly the class of pointed models accepted by A.*

Proof. We shall first establish that the algorithm defined above halts in the class of finite pointed SB(Π)-models. Assume that it does not halt in some finite model (M, w). By symmetry, we may assume that $(M, w) \models Q_1 \wedge \neg Q_2 \wedge \neg Q_3$. It is easy to see that for each $n \in \mathbb{N}$, the node w must be the first member w_0 of some finite path $(w_i)_{i \in \{0,..,n\}}$ of proper nodes that satisfy the predicates Q_i in the cyclic fashion such that $(M, w_1) \models Q_1$, $(M, w_2) \models Q_2$, $(M, w_3) \models Q_3$, $(M, w_4) \models Q_1$ and so on. Therefore, since M is a finite model, the node w must be the first member w_0 of some infinite path $(w_i)_{i \in \mathbb{N}}$ of proper nodes that satisfy the predicates Q_i in the cyclic fashion, and the path must contain a cycle. The cycle will generate a word with a cube factor that will ultimately be detected at w. Therefore the automaton at w halts. This is a contradiction.

To see that the automaton does not define a constant-time algorithm, consider for example cycle graphs with only proper nodes that satisfy the predicates Q_1, Q_2 and Q_3 in the cyclic fashion. Let ω denote the infinite Morse-Thue sequence of zeros and ones. The sequence does not contain a cube factor. For each finite prefix v of ω of some lenth $k = 3n$, where $n \in \mathbb{N}_{\geq 1}$, let $C(v)$ denote the $(\{P_0, P_1, Q_1, Q_2, Q_3\}, \{R\})$-model M such that the following conditions hold.

1. M is a (labelled) cycle graph whose domain is a cycle $(v_i)_{i \in \{0,\ldots,k-1\}}$. So $(u, v) \in R$ iff we have $\{u, v\} = \{v_i, v_{i+1}\}$ for some i or $\{u, v\} = \{v_0, v_{k-1}\}$. Each node is proper.

$(M, v_0) \models Q_1$ and $(M, v_1) \models Q_2$. (Cyclicity is not an essential point here, as we shall see.)

2. The sequence $(v_0, ...v_{k-1})$ and the predicates P_0 and P_1 define the prefix v of the Morse-Thue sequence: for each $i \in \{ 0, ..., k-1 \}$, we have $(M, v_i) \models P_x$ if $x \in \{ 0, 1 \}$ is the i-th letter of the Morse-Thue sequence (assuming the convention that the Morse-Thue sequence begins with the *zeroth* letter).

Since there are such cycle graphs $C(v)$ of arbitrarily large finite sizes, and since the Morse-Thue sequence is cube-free, it is easy to see that the automaton A is not a constant-time automaton; consider points far from v_0. By considering path graphs of different lengths that contain finite prefixes of the Morse-Thue sequence,[3] it is easy to see that there is no constant-time automaton that accepts exactly the same pointed models as A. □

4 Halting and convergence in arbitrary networks

The aim of this section is to prove Theorem 4.3 and discuss some of its corollaries.

Let Π be a finite set of unary relation symbols and $\mathcal{R} = \{ R_1, ..., R_k \}$ a finite set of binary relation symbols. The set T_0 of $(\Pi, \mathcal{R}, 0)$-*types* is defined to be the set containing a conjunction $\bigwedge_{P \in U} P \wedge \bigwedge_{P \in \Pi \setminus U} \neg P$ for each set $U \subseteq \Pi$, and no other formulae. We assume some canonical bracketing and ordering of conjuncts, so that there is exactly one conjunction for each set $U \subseteq \Pi$ in T_0. Note also that $\bigwedge \emptyset = \top$. The $(\Pi, \mathcal{R}, 0)$-type $\tau_{(M,w),0}$ of a pointed $(\Pi, \mathcal{R})$-model (M, w) is the unique formula φ in T_0 such that $(M, w) \models \varphi$.

Assume then, recursively, that we have defined the set T_n of $(\Pi, \mathcal{R}, n)$-types. Assume that T_n is finite, and assume also that each pointed $(\Pi, \mathcal{R})$-model (M, w) satisfies exactly one $(\Pi, \mathcal{R}, n)$-type $\tau_{(M,w),n}$. Define

$$\begin{aligned} \tau_{(M,w),n+1} \;:=\; & \tau_{(M,w),n} \\ & \wedge \bigwedge \{ \langle R_i \rangle \tau \mid \tau \in T_n,\ (M, w) \models \langle R_i \rangle \tau,\ i \in \{ 1, ..., k \} \} \\ & \wedge \bigwedge \{ \neg \langle R_i \rangle \tau \mid \tau \in T_n,\ (M, w) \not\models \langle R_i \rangle \tau,\ i \in \{ 1, ..., k \} \}. \end{aligned}$$

The formula $\tau_{(M,w),n+1}$ is the $(\Pi, \mathcal{R}, n+1)$-type of (M, w). We assume some standard ordering of conjuncts and bracketing, so that if two types $\tau_{(M,w),n+1}$ and $\tau_{(N,v),n+1}$ are equivalent, they are actually the same formula. We define T_{n+1} to be the set

$$\{ \tau_{(M,w),n+1} \mid (M, w) \text{ is a pointed } (\Pi, \mathcal{R})\text{-model} \}.$$

We observe that the set T_{n+1} is finite, and that for each pointed $(\Pi, \mathcal{R})$-model (M, w), there exists exactly one type $\tau \in T_{n+1}$ such that $(M, w) \models \tau$.

It is easy to show by a simple induction on modal depth that each formula φ of $\mathrm{ML}(\Pi, \mathcal{R})$ is equivalent to the disjunction of exactly all $(\Pi, \mathcal{R}, md(\varphi))$-types τ such that $\tau \models \varphi$, where $\tau \models \varphi$ means that for all pointed $(\Pi, \mathcal{R})$-models (M, w), we have $(M, w) \models \tau \Rightarrow (M, w) \models \varphi$. (Note that $\bigvee \emptyset = \bot$.)

Define $\mathcal{T} = \{ \tau \mid \tau \text{ is a } (\Pi, \mathcal{R}, n)\text{-type for some } n \in \mathbb{N} \}$. A $(\Pi, \mathcal{R})$-*type automaton* A is a $(\Pi, \mathcal{R})$-automaton whose set of states is $\mathcal{T}$. The set of messages is also the set $\mathcal{T}$. Furthermore, the initial transition function π is defined such that the state of A at (M, w)

[3] We can create such a labelled path graph by deleting the edge $v_{k-1}v_0$ of some $C(v)$. Far from v_0, a large neighbourhood will look like a midde segment of the Morse-Thue sequence. Now fiddle about with the ends.

in round $n = 0$ is the $(\Pi, \mathcal{R}, 0)$-type $\tau_{(M,w),0}$. Let $(N_1, ..., N_k)$ be a sequence of sets of $(\Pi, \mathcal{R}, n)$-types. If τ_n is a $(\Pi, \mathcal{R}, n)$-type, we define $\delta((N_1, ..., N_k), \tau_n)$ to be the type

$$\tau_n \ \wedge \ \bigwedge\{ \ \langle R_1 \rangle \tau \mid \tau \in N_1 \ \} \wedge \ \bigwedge\{ \ \neg \langle R_1 \rangle \tau \mid \tau \in T_n \setminus N_1 \ \}$$
$$\vdots$$
$$\bigwedge\{ \ \langle R_k \rangle \tau \mid \tau \in N_k \ \} \wedge \ \bigwedge\{ \ \neg \langle R_k \rangle \tau \mid \tau \in T_n \setminus N_k \ \},$$

when such a type exists. Otherwise and on other kinds of inputs, δ is defined arbitrarily. The message construction function μ is defined such that $\mu(\tau, R_i) = \tau$ for each R_i. The sets of accepting and rejecting states can be defined differently for different type automata. It is easy to see that the state of any type automaton A at (M, w) in round n is τ iff the $(\Pi, \mathcal{R}, n)$-type of (M, w) is τ.

The following proposition generalizes Theorem 4 of the article [6].

Proposition 4.1. *Let Π and $\mathcal{R}$ be finite sets of unary and binary relation symbols, respectively. Let A be a $(\Pi, \mathcal{R})$-automaton. Let $\mathcal{C}$ be the class of all pointed $(\Pi, \mathcal{R})$-models. The class $\mathcal{K} \subseteq \mathcal{C}$ of pointed models accepted by A is definable with respect to $\mathcal{C}$ by a (possibly infinite) disjunction $\bigvee \Phi$ of formulae of $\mathrm{ML}(\Pi, \mathcal{R})$, and the set $\mathcal{J} \subseteq \mathcal{C}$ of pointed models rejected by A is definable with respect to $\mathcal{C}$ by a (possibly infinite) disjunction $\bigvee \Psi$ of formulae of $\mathrm{ML}(\Pi, \mathcal{R})$. If A specifies a local algorithm in some class $\mathcal{H} \subseteq \mathcal{C}$ and $\mathcal{M} \subseteq \mathcal{H}$ is the class of pointed models accepted by A in $\mathcal{H}$, then there is a formula of $\mathrm{ML}(\Pi, \mathcal{R})$ that defines $\mathcal{M}$ with respect to $\mathcal{H}$.*

Proof. Let (M, w) be a pointed $(\Pi, \mathcal{R})$-model. Let A be $(\Pi, \mathcal{R})$-automaton. Let $n \in \mathbb{N}$. We let $A\big((M, w), n\big)$ denote the state of the automaton A at the node w in round n. We shall first show that for all $n \in \mathbb{N}$ and all pointed $(\Pi, \mathcal{R})$-models (M, w) and (N, v), if the models (M, w) and (N, v) satisfy exactly the same $(\Pi, \mathcal{R}, n)$-type, then $A\big((M, w), k\big) = A\big((N, v), k\big)$ for each $k \leq n$ and each $(\Pi, \mathcal{R})$-automaton A. We prove the claim by induction on n. For $n = 0$, the claim holds trivially by definition of the transition function π.

Let (M, w) and (N, v) be pointed $(\Pi, \mathcal{R})$-models that satisfy exactly the same $(\Pi, \mathcal{R}, n+1)$-type τ_{n+1}. Let A be an automaton and δ the transition function of A. Call $q_n = A\big((M, w), n\big)$ and $q_{n+1} = A\big((M, w), n + 1\big)$. Assume that τ_{n+1} is the formula

$$\tau_n \ \wedge \ \bigwedge\{ \ \langle R_1 \rangle \tau \mid \tau \in N_1 \ \} \wedge \ \bigwedge\{ \ \neg \langle R_1 \rangle \tau \mid \tau \in T_n \setminus N_1 \ \}$$
$$\vdots$$
$$\bigwedge\{ \ \langle R_k \rangle \tau \mid \tau \in N_k \ \} \wedge \ \bigwedge\{ \ \neg \langle R_k \rangle \tau \mid \tau \in T_n \setminus N_k \ \}.$$

The models (M, w) and (N, v) satisfy the same $(\Pi, \mathcal{R}, n+1)$-type τ_{n+1}, and therefore they must satisfy the same $(\Pi, \mathcal{R}, n)$-type τ_n. By the induction hypothesis, we therefore conclude that $A\big((N, v), n\big) = q_n$.

Let us then define that if L is the set of exactly all $(\Pi, \mathcal{R}, n)$-types τ such that $(M, w) \models \langle R_i \rangle \tau$, then L is *the set of $(\Pi, \mathcal{R}, n)$-types realized by the R_i-successors of w*.

Let $i \in \{1, ..., k\}$. Since (M, w) and (N, v) satisfy the same $(\Pi, \mathcal{R}, n + 1)$-type τ_{n+1}, the set of $(\Pi, \mathcal{R}, n)$-types realized by the R_i-successors of w is the same as the set realized by

the R_i-successors of v: that set is N_i in both cases. Therefore, by the induction hypothesis, the set of states obtained by the R_i-successors of w in round n is exactly the same as the set of states obtained by the R_i-successors of v in round n. This is true for an arbitrary $i \in \{1, ..., k\}$. Thus the k-tuple of message sets received by w in round $n+1$ is exactly the same as the k-tuple received by v. Therefore, since $A\big((N, v), n\big) = A\big((M, w), n\big) = q_n$, we conclude that $A\big((N, v), n+1\big) = q_{n+1}$, as required.

We have now established that if (M, w) an (N, v) satisfy the same $(\Pi, \mathcal{R}, n)$-type, then any automaton A produces the same state at (M, w) and (N, v) in all rounds $m \leq n$. We are ready to complete the proof of the proposition.

Let A be an arbitrary $(\Pi, \mathcal{R})$-automaton. Define

$$\mathcal{T} = \{\ \tau \mid \tau \text{ is a } (\Pi, \mathcal{R}, n)\text{-type for some } n \in \mathbb{N}\ \}.$$

Let Φ denote the set of exactly all types $\tau \in \mathcal{T}$ such that for some n, the type τ is the $(\Pi, \mathcal{R}, n)$-type of some pointed $(\Pi, \mathcal{R})$-model (M, w), and furthermore, the automaton A accepts (M, w) in round n. Define the possibly infinite disjunction $\bigvee \Phi$. We shall establish that for all pointed $(\Pi, \mathcal{R})$-models (M, w), we have $(M, w) \models \bigvee \Phi$ iff A accepts (M, w).

Assume that $(M, w) \models \bigvee \Phi$. Thus $(M, w) \models \tau_n$ for some $(\Pi, \mathcal{R}, n)$-type τ_n of some pointed model (M', w') accepted by A in round n. Now (M, w) and (M', w') satisfy the same $(\Pi, \mathcal{R}, n)$-type τ_n, so A produces exactly the same state at (M, w) as at (M', w') in all rounds $l \leq n$. Therefore (M, w) must be accepted by A in round n.

Assume that (M, w) is accepted by the automaton A. The pointed model (M, w) is accepted in some round n, and thus the $(\Pi, \mathcal{R}, n)$-type of (M, w) is one of the formulae in Φ. Therefore $(M, w) \models \bigvee \Phi$.

Let $\mathcal{C}$ be the class of exactly all pointed $(\Pi, \mathcal{R})$-models and $\mathcal{K} \subseteq \mathcal{C}$ the class of pointed models accepted by A. We have established that $\bigvee \Phi$ defines the class $\mathcal{K} \subseteq \mathcal{C}$ with respect to $\mathcal{C}$. Let $\mathcal{J} \subseteq \mathcal{C}$ be the class of pointed models rejected by A. Let Ψ be the set of types $\tau \in \mathcal{T}$ such that for some n, the type τ is the $(\Pi, \mathcal{R}, n)$-type of some pointed $(\Pi, \mathcal{R})$-model (M, w), and furthermore, the automaton A rejects (M, w) in round n. By an argument practically identical to the one above establishing that $\mathcal{K}$ is definable by $\bigvee \Phi$, one can establish that $\bigvee \Psi$ defines the class $\mathcal{J}$ with respect to $\mathcal{C}$.

If A defines a local algorithm in some class $\mathcal{H} \subseteq \mathcal{C}$, then for some $l \in \mathbb{N}$, each pointed model in $\mathcal{H}$ is either accepted or rejected in some round $l' \leq l$. Let $\mathcal{H}' \subseteq \mathcal{H}$ be the class of pointed models accepted by A in $\mathcal{H}$. Define Γ to the set of types $\tau \in \mathcal{T}$ such that τ is the $(\Pi, \mathcal{R}, n)$-type of some $(M', w') \in \mathcal{H}$ accepted in round n. The set Γ is finite, and the argument above shows that $\bigvee \Gamma$ defines $\mathcal{H}'$ with respect to $\mathcal{H}$. □

Theorem 4.2 (Compactness Theorem, see [9])**.** *Assume T is a set of formulae of* $\mathrm{FO}(\Pi, \mathcal{R})$ *such that for each finite subset T' of T, there exists a $(\Pi, \mathcal{R})$-interpretation (M, f) such that $(M, f) \models T'$. Then there exists a $(\Pi, \mathcal{R})$-interpretation (M, f) such that $(M, f) \models T$.*

It is an immediate consequence of the compactness theorem that if $T \models \varphi$, then there is a finite subset T' of T such that $T' \models \varphi$.

Theorem 4.3. *Let Π and $\mathcal{R}$ be finite sets of unary and binary relation symbols. Let $\mathcal{C}$ be the class of all pointed $(\Pi, \mathcal{R})$-models. Let $\mathcal{H} \subseteq \mathcal{C}$ be a class definable by a first-order $(\Pi, \mathcal{R})$-sentence. If a $(\Pi, \mathcal{R})$-automaton converges in $\mathcal{H}$, then it specifies a local algorithm in $\mathcal{H}$.*

Proof. Assume a $(\Pi, \mathcal{R})$-automaton A converges in $\mathcal{H} \neq \emptyset$. Let $\mathcal{K} \subseteq \mathcal{H}$ be the class of pointed models accepted by A in $\mathcal{H}$. By the proof of Proposition 4.1, there is a disjunction $\bigvee \Phi$ of types that defines $\mathcal{K}$ with respect to $\mathcal{H}$ and a disjunction $\bigvee \Psi$ of types that defines $\mathcal{H} \setminus \mathcal{K}$

with respect to $\mathcal{H}$. The $(\Pi, \mathcal{R}, n)$-type of a pointed $(\Pi, \mathcal{R})$-model $(M, w) \in \mathcal{H}$ is in Φ iff the automaton A accepts (M, w) in round n. Similarly, the $(\Pi, \mathcal{R}, n)$-type of $(N, v) \in \mathcal{H}$ is in Ψ iff the automaton A rejects (N, v) in round n.

Let ψ be a first-order sentence that defines the class $\mathcal{H}$. Let $T = \{ \neg St_x(\varphi) \mid \varphi \in \Phi \}$ and $Y = \{ \neg St_x(\varphi) \mid \varphi \in \Psi \}$. Since $\bigvee \Phi$ defines $\mathcal{K}$ with respect to $\mathcal{H}$ and $\bigvee \Psi$ defines $\mathcal{K} \setminus \mathcal{H}$ with respect to $\mathcal{H}$, we have $T \cup Y \cup \{ \psi \} \models \bot$. By the compactness theorem, there is a finite set $U \subseteq T \cup Y \cup \{ \psi \}$ such that $U \models \bot$. Let $S = U \cap T$ and $X = U \cap Y$. Now, define $S^* = \{ \varphi \in \mathrm{ML}(\Pi, \mathcal{R}) \mid St_x(\varphi) \in S \}$. Define X^*, T^* and Y^* analogously. We shall next establish that $\bigwedge X^*$ defines $\mathcal{K}$ with respect to $\mathcal{H}$.

Assume $(M, w) \in \mathcal{K}$. Thus $(M, w) \models Y^*$, and hence $(M, w) \models \bigwedge X^*$. Assume then that $(N, v) \in \mathcal{H} \setminus \mathcal{K}$. Therefore $(N, v) \models T^*$ (and thus $(N, v) \models S^*$). Since $(N, v) \in \mathcal{H}$, we have $N \models \psi$. Now assume, for the sake of contradiction, that $(N, v) \models \bigwedge X^*$. Therefore $(N, f[x \mapsto v]) \models T \cup X \cup \{ \psi \}$, whence $(N, v) \models \bot$ (since $U \models \bot$). This is a contradiction.

We then establish that $\bigwedge S^*$ defines $\mathcal{H} \setminus \mathcal{K}$ with respect to $\mathcal{H}$. Assume $(M, w) \in \mathcal{H} \setminus \mathcal{K}$. Thus $(M, w) \models T^*$, and hence $(M, w) \models \bigwedge S^*$. Assume then that $(N, v) \in \mathcal{K}$. Therefore $(N, v) \models Y^*$ (and thus $(N, v) \models X^*$). Since $(N, v) \in \mathcal{H}$, we have $N \models \psi$. Now assume, for the sake of contradiction, that $(N, v) \models \bigwedge S^*$. Therefore $(N, f[x \mapsto v]) \models S \cup Y \cup \{ \psi \}$, whence $(N, v) \models \bot$ (since $U \models \bot$.) This is a contradiction.

The finite sets X^* and S^* are negations of types. Let Ψ' be the set of types whose negations are in X^* and Φ' the set of types whose negations are in S^*. Notice indeed that $\Psi' \subseteq \Psi$ and $\Phi' \subseteq \Phi$. The disjunction $\bigvee \Phi'$ defines $\mathcal{K}$ with respect to $\mathcal{H}$, and the disjunction $\bigvee \Psi'$ defines $\mathcal{H} \setminus \mathcal{K}$ with respect to $\mathcal{H}$. Let n be the greatest integer j such that there is a $(\Pi, \mathcal{R}, j)$-type in $\Phi' \cup \Psi'$. For each pointed model (M, w) in $\mathcal{H}$, the automaton A either accepts or rejects (M, w) in some round $m \leq n$. To see this, let $(M, w) \in \mathcal{K}$. Thus $(M, w) \models \bigvee \Phi'$, so $(M, w) \models \tau$ for some $(\Pi, \mathcal{R}, i)$-type τ, where $i \leq n$. Since $\Phi' \subseteq \Phi$, we have $\tau \in \Phi$, and this means that (M, w) is accepted in round i by A. The argument is similar when $(M, w) \in \mathcal{H} \setminus \mathcal{K}$. Therefore A specifies a local algorithm in $\mathcal{H}$. □

As we saw in Section 2, each class $\mathcal{PN}(n)$ is definable by a related first-order sentence $\varphi_{\mathrm{PN}(n)}$. Hence all halting algorithms in the port-numbering model are local algorithms. This is the main corollary of the investigations in this section. In Section 3 we saw that finiteness gives rise to halting non-local behaviour. What kinds of other non-first-order properties are there that lead to the existence of non-local halting algorithms?

5 Concluding considerations

One of the immediate observations based on the above investigations is that Theorem 4.3 can surely be generalized in a range of interesting canonical ways. One intriguing idea that should be pursued involves considering novel classes of automata whose behaviour can be captured by modal logics with (first-order definable) *generalized modalities*. Graded modalities are an example of generalized modalities, but only an example. One can define, for example that, $(M, w) \models \langle R, S \rangle (P, Q)$ iff $(M, f[x \mapsto w]) \models \exists y \exists z \exists v \big(R(x, y) \wedge S(y, z) \wedge P(z) \wedge Q(v) \big)$, or whatever.

There is no essential reason to limit attention to modalities that deal only with binary and unary relations and output sets. However, let us consider modalities that satisfy such limitations. Each such modality is associated with a *width* $(n, m) \in \mathbb{N} \times \mathbb{N}$, where n is the number of input unary relations and m the number of input binary relations. For example, the width of the formula $\langle R, S \rangle (P, Q)$ above is $(2, 2)$, and the width of the ordinary Kripke formula $\langle R \rangle P$ is $(1, 1)$. Formally, a modality of width (n, m) is a class function F (too large

to be a set) that maps any structure $(W, P_1, ..., P_n, R_1, ..., R_m)$ to a subset of the domain W; here P_i are unary and R_i binary relations. The operator F satisfies the constraint that if f is an isomorphism from $(W, P_1, ..., P_n, R_1, ..., R_m)$ to $(U, P'_1, ..., P'_n, R'_1, ..., R'_m)$, then $f(F(\,(W, P_1, ..., P_n, R_1, ..., R_m)\,)\,) = F(\,(U, P'_1, ..., P'_n, R'_1, ..., R'_m)\,)$. The logic that deals with such modalities can be based on a grammar of the type

$$\varphi \ ::= \ \top \mid P \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \langle S_{i_1}, ..., S_{i_m}\rangle(\varphi_1, ..., \varphi_n),$$

where $P \in \Pi$ are unary and $S_{i_1}, ..., S_{i_m} \in \mathcal{R}$ binary relation symbols; here the models considered are pointed $(\Pi, \mathcal{R})$-models. The semantics asserts that $(M, w) \models \langle S_{i_1}, ..., S_{i_m}\rangle(\varphi_1, ..., \varphi_n)$ iff we have

$$w \in F\big(\, Dom(M), ||\varphi_1||^M, ..., ||\varphi_n||^M, S^M_{i_1}, ..., S^M_{i_m})\,\big),$$

where $||\varphi_i||^M = \{\ w \in Dom(M) \mid (M, w) \models \varphi_i\ \}$.

Notice that modalities do not have to be local in any sense. But of course one can define several interesting classes of local modalities. Define a *contact modality* of width (n, m) to be an isomorphically closed class $\mathcal{C}$ of structures $(W, c, P_1, ..., P_n, R_1, ..., R_m)$, where c is a constant, P_i are unary and R_i binary relations. The related language can be based on a grammar of the type given above. In order to define the semantics, let (M, w) be a pointed $(\Pi, \mathcal{R})$-model, where $\mathcal{R} = \{S_1, ..., S_l\}$. Let us define the semantics for a formula $\langle S_{i_1}, ..., S_{i_m}\rangle(\varphi_1, ..., \varphi_n)$. Let $n(w)$ denote the set

$$\{\ u \in Dom(M) \mid (w, u) \in S^M_{i_1} \cup ... \cup S^M_{i_m} \text{ or } (u, w) \in S^M_{i_1} \cup ... \cup S^M_{i_m}\ \}.$$

Call $B = n(w) \cup \{w\}$. The semantics now dictates that $(M, w) \models \langle S_{i_1}, ..., S_{i_m}\rangle(\varphi_1, ..., \varphi_n)$ iff

$$\big(B,\, w,\, ||\varphi_1||^M \cap B,\, ...\,,\, ||\varphi_n||^M \cap B,\, S^M_{i_1} \restriction B\,, ...\,,\, S^M_{i_m} \restriction B\,\big) \in\ \mathcal{C},$$

where $S^M_{i_j} \restriction B$ is the restriction of the relation $S^M_{i_j}$ to the set B. So, essentially the formula only scans a local neigbourhood of the evaluation point w (and thus there can be a lot of junk in $\mathcal{C}$; it is easy to work out a cleaner definition of the same class of operators, if desired).

Define a *successor modality* of width $(n, 1)$ to be an isomorphically closed class $\mathcal{C}$ of structures $(W, P_1, ..., P_n)$. Here P_i are unary relations. The related language is given by the grammar

$$\varphi \ ::= \ \top \mid P \mid \neg\varphi \mid (\varphi_1 \wedge \varphi_2) \mid \langle S\rangle(\varphi_1, ..., \varphi_n),$$

where $P \in \Pi$ are unary and $S \in \mathcal{R}$ binary relation symbols. Let (M, w) be a pointed $(\Pi, \mathcal{R})$-model, and let $S \in \mathcal{R}$. Consider a formula $\langle S\rangle(\varphi_1, ..., \varphi_n)$. Let B denote the set

$$\{\ u \in Dom(M) \mid (w, u) \in S^M\ \}.$$

The semantics now dictates that $(M, w) \models \langle S\rangle(\varphi_1, ..., \varphi_n)$ iff

$$\big(B,\, ||\varphi_1||^M \cap B,\, ...\,,\, ||\varphi_n||^M \cap B\big) \in \mathcal{C}.$$

In addition to concrete automata, one can consider abstract devices that scan—in one way or another— the model they sit in (or on). Let $(\Pi, \mathcal{R})$ be a pair of finite sets of unary and binary relation symbols, respectively. Let $l \in \mathbb{N}$ and let F be the set of finite and countably infinite disjunctions of first-order $(\Pi, \mathcal{R})$-formulae $\varphi(x_1, ..., x_l)$ with exactly the l free variables $x_1, ..., x_l$. Let $f : \mathbb{N} \to \mathrm{F}$ and $g : \mathbb{N} \to \mathrm{F}$ be functions. Let M be a $(\Pi, \mathcal{R})$-model and $\overline{v} = (v_1, ..., v_l)$ an l-tuple of elements in the domain of M. We say that (f, g) accepts $(M, \overline{v})$ in round n if the following conditions hold.

1. $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models f(n)$.

2. For all $m < n$, $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \not\models f(m)$.

3. For all $m \leq n$, $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \not\models g(m)$.

We say that (f, g) rejects $(M, \overline{v})$ in round n if (g, f) accepts $(M, \overline{v})$ in round n. We say that (f, g) converges on $(M, \overline{v})$ if (f, g) either accepts or rejects $(M, \overline{v})$ in some round $n \in \mathbb{N}$.

Let $\mathcal{C}$ be some class of structures $(M, \overline{v})$, where $\overline{v}$ is an l-tuple of elements in the domain of the $(\Pi, \mathcal{R})$-model M. We say that (f, g) converges in $\mathcal{C}$ if it converges on every structure in $\mathcal{C}$. We say that the pair (f, g) is local in $\mathcal{C}$ if there exists an $n \in \mathbb{N}$ such that for each structure $(M, \overline{v})$ in $\mathcal{C}$, the pair (f, g) either accepts of rejects $(M, \overline{v})$ in some round $m \leq n$. Assume there exist some n and m and some $(M, \overline{v}) \in \mathcal{C}$ such that $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models f(n)$ and $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models g(m)$. Then we say that (f, g) is *not refining* in $\mathcal{C}$. Otherwise (f, g) is *refining* in $\mathcal{C}$.

We say that $\mathcal{C}$ is definable by a first-order theory if there exists a possibly infinite set B of first-order $(\Pi, \mathcal{R})$-formulae $\varphi(x_1, ..., x_l)$ with exactly the l free variables $x_1, ..., x_l$ such that for each $(\Pi, \mathcal{R})$-model M and each l-tuple $\overline{v}$ of elements in the domain of M, we have

$$(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models \bigwedge B \text{ iff } (M, \overline{v}) \in \mathcal{C}.$$

Theorem 5.1. *Assume $\mathcal{C}$ is definable by a first-order theory. Then, if (f, g) is refining and converges in $\mathcal{C}$, it is local in $\mathcal{C}$.*

Proof. Assume (f, g) is refining and converges in $\mathcal{C} \neq \emptyset$. Call $\Phi = \{ \varphi \mid \varphi \in f(n) \text{ for some } n \}$, where $\varphi \in f(n)$ means that φ is one of the disjuncts of $f(n)$. Similarly, let $\Psi = \{ \varphi \mid \varphi \in g(n) \text{ for some } n \}$. Let $T = \{ \neg\varphi \mid \varphi \in \Phi \}$ and $Y = \{ \neg\varphi \mid \varphi \in \Psi \}$. Let the theory H define $\mathcal{C}$. Since (f, g) converges in $\mathcal{C}$, we have $T \cup Y \cup H \models \bot$. By the compactness theorem, there is a finite set $U \subseteq T \cup Y \cup H$ such that $U \models \bot$. Let $S = U \cap T$ and $X = U \cap Y$.

Next we show that X defines with respect to $\mathcal{C}$ the class of structures $(M, \overline{v}) \in \mathcal{C}$ accepted by (f, g), and that S defines with respect to $\mathcal{C}$ the class of structures in $\mathcal{C}$ rejected by (f, g).

Assume $(M, \overline{v}) \in \mathcal{C}$ is accepted by (f, g). As (f, g) is refining in $\mathcal{C}$, thus $(M, \overline{v}) \models Y$, and hence $(M, \overline{v}) \models X$. Here $(M, \overline{v}) \models Y$ means that $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models \bigwedge Y$, and obviously similarly for X.

Assume then that $(N, \overline{u}) \in \mathcal{C}$ is not accepted by (f, g). Thus $(N, \overline{u})$ is rejected by (f, g). As (f, g) is refining in $\mathcal{C}$, therefore $(N, \overline{u}) \models T$ (and thus $(N, \overline{u}) \models S$). Since $(N, \overline{u}) \in \mathcal{C}$, we have $(N, \overline{u}) \models H$. Now assume, for the sake of contradiction, that $(N, \overline{u}) \models X$. Therefore $(N, \overline{u}) \models T \cup X \cup H$, whence $(N, \overline{u}) \models \bot$ (since $U \models \bot$). This is a contradiction. Therefore X defines with respect to $\mathcal{C}$ the class of structures $(M, \overline{v})$ in $\mathcal{C}$ accepted by (f, g).

Assume $(M, \overline{v}) \in \mathcal{C}$ is rejected by (f, g). As (f, g) is refining in $\mathcal{C}$, thus $(M, \overline{v}) \models T$, and hence $(M, \overline{v}) \models S$. Assume then that $(N, \overline{u}) \in \mathcal{C}$ is not rejected by (f, g). Therefore $(N, \overline{u}) \models Y$ (and thus $(N, \overline{u}) \models X$). Since $(N, \overline{u}) \in \mathcal{C}$, we have $(N, \overline{u}) \models H$. Now assume, for the sake of contradiction, that $(N, \overline{u}) \models S$. Therefore $(N, \overline{u}) \models S \cup Y \cup H$, whence $(N, \overline{u}) \models \bot$. This is a contradiction. Therefore S defines with respect to $\mathcal{C}$ the class of structures $(M, \overline{v})$ in $\mathcal{C}$ not accepted by (f, g).

Call $\Psi' = \{ \varphi \mid \neg\varphi \in X \}$ and $\Phi' = \{ \varphi \mid \neg\varphi \in S \}$. Notice that $\Psi' \subseteq \Psi$ and $\Phi' \subseteq \Phi$. Notice also that by the above arguments concerning X and S, the finite disjunction $\bigvee \Phi'$ defines with respect to $\mathcal{C}$ the class of structures in $\mathcal{C}$ accepted by (f, g), and the finite disjunction $\bigvee \Psi'$ defines with respect to $\mathcal{C}$ the class of structures in $\mathcal{C}$ not accepted by (f, g). Therefore, for all $(M, \overline{v}) \in \mathcal{C}$, we have either $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models \bigvee \Phi'$ or $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models \bigvee \Psi'$.

Let n be the greatest integer i such that there exists a formula $\varphi \in \Phi' \cup \Psi'$ satisfying the following conditions.

1. φ is a disjunct of $f(i)$ or $g(i)$.

2. If φ is a disjunct of $f(i)$, then for all $j < i$, the formula φ is not a disjunct of $f(j)$. Similarly, if φ is a disjunct of $g(i)$, then for all $j < i$, the formula φ is not a disjunct of $g(j)$.

Each formula in $\Phi' \cup \Psi'$ satisfies the above two conditions for exactly one $j \leq n$. Therefore, for an arbitrary structure $(M, \overline{v}) \in \mathcal{C}$ accepted by (f, g), since we have $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models \bigvee \Phi'$, we must also have $(M, [x_1 \mapsto v_1, ..., x_l \mapsto v_l]) \models f(m)$ for some $m \leq n$. For a structure $(N, \overline{u}) \in \mathcal{C}$ not accepted by (f, g), since $(N, [x_1 \mapsto u_1, ..., x_l \mapsto u_l]) \models \bigvee \Psi'$, we have $(N, [x_1 \mapsto u_1, ..., x_l \mapsto u_l]) \models g(k)$ for some $k \leq n$. Therefore (f, g) specifies a local algorithm in $\mathcal{C}$. □

Note that in the finite, a countably infinite disjunction (conjunction) of first-order formulae can define any isomorphically closed class of models of a finite vocabulary. Therefore, *in the finite*, the framework with computation systems (f, g) and first-order theory definable classes of models in a sense *contains* a rather extremely comprehensive variety of distributed computation models. Theorem 5.1 shows that if infinite models are included in the picture, convergence of a refining (f, g) implies locality on first-order theory definable classes. Of course here the way infinite models are included into the picture is the way the systems (f, g) deal with infinite models. Even if a system S is in the finite contained in a system S' based on devices (f, g), it is possible that there is some canonical way of extending the scope of S to include infinite models, and the resulting system is no longer contained in the system S'.

Is Theorem 5.1 surprising? Not necessarily. The equation $F = ma$ of classical physics is surprising. It states that $F = ma$, not $F = ma^r$ with some random exponent r. The definition that $F = ma^r$ or $F = m^r a$, with $r \neq 1$, would not do too well. To see this, place two objects A and B connected by a thin piece of thread on two tilted slopes that face each other. A hill with A and B on opposite slopes and connected via the top of the hill will do. By using homogenous pieces of matter, one can be reasonably sure that the mass m_A of A is more or less twice the mass m_B of B. By using smooth surfaces or light trolleys, frictional effects can be made negligible. Now, adjust the angles of the slopes such that the acceleration a_B of B (due to gravity) along the slope of B is twice the the acceleration a_A of A along the slope of A. Now $m_A a_A = m_B a_B$, so the system is stationary. It would not be stationary if F was $m^r a$ with some sufficiently ugly r. This setting somehow seems to directly suggest that the quantities of mass, length and time, which superficially *may appear* independent of each other, are after all neatly connected and dependent on each other in some way. That is surprising. Innit? Right, is it? (Ha ha!)

Often one sees it first stated that *it can be shown that* every nonempty subset of $\mathbb{N}$ has a smallest number, indicating that this requires a proof, but still for example a transition from Dedekind infinity to infinity in terms of $\mathbb{N}$ is later on made *without* stating that *it can be shown that* blah blah blah. So the transition then does not require a proof. That is surprising. Or perhaps it is random. It would be best not to implicitly claim the use of an axiomatic approach where it is not really employed. For another example, it is can be strange to first state here and there that the Axiom of Choice is employed, but still then go without comment from Dedekind infinity to infinity in terms of $\mathbb{N}$. Also, works that claim an axiomatic approach but still specify neither a notion of a model nor a formal language, can sometimes appear strange.

Writings stating that some informally specified mathematical theorem T is independent of ZFC without any effort to try to explain which first-order $\{\in\}$-formula is meant to correspond to T, can sometimes be surprising. We may be able to conclude that some Turing machine TM halts simply by looking at its specification, but whether the non-halting of TM is independent

of ZFC or a theorem of ZFC, depends on how we encode halting statements of Turing machines in first-order logic. In the absence of the Axiom of Choice, Dedekind inifinity and infinity in terms of $\mathbb{N}$ provide a simple example of an informal concept formulated in two non-equivalent ways. Its a wide world, innit? Perhaps Theorem 5.1 is surprising, perhaps not. But the relevance of the above three paragraphs to this article is surprising, innit?

Ok, that is enough cheeky stuff for now. We have looked at rather general frameworks where one can define logics that correspond to message passing automata and related devices. Let us then briefly have a look at a rather general framework for defining complexity classes for the port-numbering model (see [12]) and for computation in automata networks. (Local finiteness is assumed, i.e., all degrees of all nodes are finite.)

Let $\mathcal{S}$ be a countably infinite set of symbols. Consider words $w, v \in \mathcal{S}^*$. Assume there exists a bijection $f : \mathcal{S} \to \mathcal{S}$ (a renaming bijection) such that we have $v = f(w_1) \cdot ... \cdot f(w_k)$; here $w = w_1 \cdot ... \cdot w_k$, where the symbols w_i are elements of $\mathcal{S}$ and $\cdot$ the concatenation symbol. Then v is a *renaming* of w specified by the bijection f. We write $v = f(w)$.

Consider words $u, t \in \mathcal{S}^*$. Assume that $u = u_1 \cdot ... \cdot u_k$ and that there exists a bijection $p : \{\, 1, ..., k \,\} \times \{\, 1, ..., k \,\}$ (a reordering bijection) such that $t = u_{p(1)} \cdot ... \cdot u_{p(k)}$. Then t is a *reordering* of u. We write $t = p(u)$. (Below it will be known from the context whether a renaming bijection or a reordering bijection is applied.)

Notice also that of course the empty word is a renaming of exactly itself, and a reordering of exactly itself.

Let $E_k \subseteq \mathcal{S}^k \times \mathcal{S}^k$ be an equivalence relation on words of some length k such that the following conditions hold.

1. If uE_kv and $f : \mathcal{S}^* \to \mathcal{S}^*$ is a renaming bijection, then $f(u)E_kf(v)$.
2. Let $p : \{1, ..., k\} \to \{1, ..., k\}$ a reordering bijection. If uE_kv, then $p(u)\, E_k\, p(v)$.

Let us say that a union $E = \bigcup_{k \in \mathbb{N}} E_k$ of such equivalence relations E_k is an *invariant equivalence relation*. With invariant equivalence relations E we can define a reasonably natural classification of input recognition capacities for message passing automata. This way it is possible to generalize the framework that leads to the different complexity classes SB, MB and VB defined in [12]. (Note that in [12], automata know the degree of the node they sit in, and two automata sitting in nodes of different degrees can behave very differently from the beginning of the computation process.) In the presence of a port numbering (or a local ordering of communication channels), at each round an automaton receives a word of messages: if there are three active input ports 1, 2 and 3, then the input word received is abc, when the messages from ports 1, 2 and 3 are a, b and c, respectively. Each invariant equivalence relation $E \subseteq \mathcal{S}^* \times \mathcal{S}^*$ defines the class of automata that cannot distinguish between E-equivalent words. The invariance condition 1 above reflects the fact that automata classes should not be sensitive to particularities related to appearances of individual messages. Similarly, the invariance condition 2 reflects the fact that automata classes should not be sensitive to particularities related to port number indices. Logical constants, such as generalized quantifiers, are usually best defined in a similar spirit of invariance.

How about output construction capacities? One could define, for example, that an acceptable collection of all possible output words of a class of automata is any set $L \subseteq \mathcal{S}^*$ such that L is closed under renaming and reordering. (If automata were strictly required to be able to produce *some* input word, one could add the condition that L contains a word of each length.) Each pair (E, L), where E is an invariant equivalence relation, would then give rise to a complexity class EL. This is one natural way of defining a general class of complexity classes and generalizing the classification of [12].

Different kinds of message passing automata networks are everywhere. They are of course essential in distributed computing, but that is not the whole story. The brain is a distributed message passing system (probably without ID numbers, at least in the case of most people). Much of the modelling of for example climate systems and stellar atmospheres is based on discrete grid-point structures (with locally communicating grid-points), often with interesting topologies. In digital physics, systems such as the universe are considered to essentially *be* computation devices, such as cellular automata. Systems are modelled as if they were computation devices. It makes obvious sense to look at the mathematical phenomena related to message passing systems even on a rather formal level. This is where logic can help.

References

[1] J-P. Allouche and J. Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, 2003.

[2] D Angluin. Local and global properties in networks of processors. In *Proc. 12th Annual ACM Symposium on Theory of Computing* (STOC), 1980.

[3] C. Areces and B. ten Cate. Hybrid Logics. In P. Blackburn, F. Wolter and J. van Benthem, (eds.), *Handbook of Modal Logic*, Elsevier, 2006.

[4] J. van Benthem and P. Blackburn. Modal Logic: a Semantic Perspective. In P. Blackburn, F. Wolter and J. van Benthem, (eds.), *Handbook of Modal Logic*, Elsevier, 2006.

[5] J. van Benthem, P. Blackburn and F. Wolter, editors. *Handbook of Modal Logic*. Elsevier Science Inc., 2006.

[6] A. Kuusisto. Modal logic and distributed message passing automata. In *Proceedings of* CSL, 2013.

[7] P. Blackburn, M. de Rijke and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.

[8] H. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer, 2nd edition, 2005.

[9] H. Ebbinghaus, J. Flum and W. Thomas. *Mathematical Logic*. Springer, 1994.

[10] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[11] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela and J. Virtema. Weak models of distributed computing, with connections to modal logic. In *Proc. 31st ACM Symposium on Principles of Distributed Computing* (PODC), 2012.

[12] L. Hella, M. Järvisalo, A. Kuusisto, J. Laurinharju, T. Lempiäinen, K. Luosto, J. Suomela and J. Virtema. Weak models of distributed computing, with connections to modal logic. Extended version, arXiv:1205.2051, 2012.

[13] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[14] J. Suomela. Survey of local algorithms. *ACM Computing Surveys* 45, 2013.